Evaluation du signe et calcul du déterminant d'une matrice à coefficients entiers

Gilles Villard

Laboratoire Lip
46, Allée d'Italie, 69364 Lyon Cedex 07, France
Cnrs - Ens Lyon - Inria
Gilles.Villard@ens-lyon.fr
http://www.ens-lyon.fr/~gvillard

Résumé

Nous introduisons à une liste de références autour de méthodes pour l'évaluation du signe et pour le calcul du déterminant d'une matrice entière. Nous nous intéressons aux complexités binaires théoriques des algorithmes en fonction de la dimension n de la matrice en entrée et du nombre de bits β nécessaires à coder ses coefficients. Deux grandes approches se dégagent avec des calculs basés sur une arithmétique de nombres flottants pour un calcul du signe seul ou sur une arithmétique exacte.

1 Introduction

Pour un «embryon de survol» des méthodes d'évaluation du signe et du calcul du déterminant, nous introduisons ici à une liste de références sur le sujet. Cette liste, forcément incomplète en particulier sur les aspects calcul flottant ou quant aux techniques de géométrie algorithmique, devrait s'étoffer. Pour une matrice $A \in \mathbb{Z}^{n\times n}$ dont les coefficients sont plus petits en valeur absolue que 2^β, nous allons reporter les complexités binaires *i.e.* en nombre d'opérations sur les bits comme fonction de n et de β.

En complexité arithmétique - nombre d'opérations sur les entiers (sans taille) - le lecteur peut se référer à Baur et à Strassen pour le lien entre calcul du déterminant et multiplication de matrices [24, 25, 5]. Voir aussi le lien avec la puissance de matrice et la classe de complexité GapL d'après Toda, Vinay, Damm et Valiant comme décrit par exemple dans [2].

Pour les matrices entières, on peut dégager deux directions de travail *a priori* : le calcul du signe du déterminant et le calcul du déterminant lui-même (le signe s'en déduit !). Nous allons comparer leurs complexités respectives. La première question peut sembler plus simple. Nous aborderons une discussion sur ce point en section 7 après avoir vu les différentes méthodes de connues dans les sections précédentes.

On notera $\mathcal{M}(n)$ le nombre d'opérations arithmétiques d'un produit de matrices, on pourra simplement considérer $\mathcal{M}(n) = O(n^3)$ avec l'algorithme standard ou $\mathcal{M}(n) = O(n^{2.376})$ avec un algorithme asymptotiquement rapide. On utilisera aussi $\mathcal{P}(l)$ pour le coût binaire de la mutiplication de deux entiers de longueur l avec $\mathcal{P}(l) = O(l^2)$ ou $\mathcal{P}(l) = O\tilde{}(l)$ en algorithmique rapide. Pour tout exposant e_1, $O\tilde{}(n^{e_1})$ signifie $O(n^{e_1}\log^{e_2} n)$ pour un certain e_2. Par défaut, on utilisera l'algorithmique «lente» pour le produit de matrices et l'algorithmique «rapide» pour le produit de nombres.

La complexité binaire du calcul du signe du déterminant sera $\mathcal{S}_{\det}(n,\beta)$, celle du calcul du déterminant lui-même sera $\mathcal{D}(n,\beta)$. On a donc $\mathcal{S}_{\det}(n,\beta) \leq \mathcal{D}(n,\beta)$. Pour les algorithmes adaptatifs (voir la Définition 3.1) ces fonctions dépendront aussi – implicitement – de la matrice A et pas seulement de sa dimension et de sa taille.

2 Résultats classiques

La donnée qui détermine en bonne partie les complexités est la taille du déterminant. Avec $\| A \|_\infty \leq 2^\beta$, on sait par l'inégalité d'Hadamard [15, Theorem 16.6] (les logarithmes seront tous à

Congrès international *Algèbre Linéaire et Arithmétique : Calcul Numérique, Symbolique et Parallèle*, 28-31 Mai 2001, Faculté des Sciences, Rabat, Maroc.

base deux) que

$$\log|\det A| \le (n/2)\log n + n\beta,$$

c'est-à-dire qu'au pire, le déterminant nécessite $O\tilde{}(n\beta)$ bits. De là, une élimination de Gauss avec taille des entiers contrôlée à la Bareiss [4] ou plusieurs éliminations de Gauss pour une arithmétique multi-modulaire, jointes au théorème chinois, permettent de calculer le déterminant exactement. Le coût «classique» du déterminant entier correspondant est (voir [15, Chap. 5]) :

$$\mathcal{D}(n,\beta) \le O\tilde{}(n^4\beta + n^3\beta^2), \tag{1}$$

ou, avec produit rapide de matrices :

$$\mathcal{D}(n,\beta) \le O\tilde{}(n\mathcal{M}(n)\beta + n^3\beta^2) \le O(n^{3.376}\beta + n^3\beta^2).$$

Une analyse fine de la borne d'Hadamard a été proposée par Abbott *et al.* [1].

Remarque 2.1 *Un sous-problème est de déterminer si une matrice est inversible ou pas – si le déterminant est nul ou pas ? En choisissant un nombre premier au hasard et en calculant le déterminant de A modulo p on peut se persuader de la réponse. Ceci peut fournir un algorithme probabiliste de type Monte Carlo de coût $O\tilde{}(n^3\log\beta + n^2\beta)$ pour le test d'égalité à zéro. Sur ce point on se reportera aussi aux algorithmes probabilistes de la section 4.*

3 Signe du déterminant : l'algorithme de Clarkson et la méthode de réseau

La détermination du signe d'une expression algébrique, et en particulier du déterminant, joue un rôle important en géométrie algorithmique. En liaison avec le déterminant, le lecteur pourra se reporter aux bibliographies de [8, 9, 6, 7, 22].

Dans le but de tirer parti, au mieux, de la rapidité des arithmétiques de nombres flottants (comparées aux arithmétiques exactes), Clarkson [10, 8] propose une méthode numérique en deux temps. La première étape, à partir de $A = (a_1, a_2, \ldots, a_n)$, consiste à construire itérativement une matrice $B = (b_1, b_2, \ldots, b_n)$ «plus orthogonale» que A et de même signe de déterminant. Grâce à de bonnes propriétés de la matrice B construite (notamment un faible défaut d'orthogonalité (2)), le signe du déterminant est ensuite calculé par élimination de Gauss sur B. Pour conduire la première phase, il suffit de calculer sur $\beta + O(n)$ bits. Le coût arithmétique dépend d'un défaut d'orthogonalité de A défini par

$$\Delta(A) = \frac{\mathrm{vol}(a_1, \ldots a_n)}{\prod_{i=1}^{n} \| a_i \|}. \tag{2}$$

Si A est inversible ce défaut vérifie :

$$1 \le \log\frac{1}{\Delta(A)} \le \prod_{i=1}^{n} \| a_i \| \le \log\left(n^{n/2} 2^{\beta n}\right) \le (n/2)\log n + n\beta$$

et cette dernière quantité borne le nombre d'itérations de la première phase. Le coût est alors donné par

$$\mathcal{S}_{\det}(n,\beta) \le O\tilde{}((n^3 + n^2\log\frac{1}{\Delta(A)}) \cdot (n+\beta)). \tag{3}$$

Rappelons que d'après la Remarque 2.1, l'inversibilité peut se tester facilement. Dans le cas général, en reposant sur les remarques de Brönnimann et Yvinec [8, 9] pour le cas où A est singulière, le coût binaire est borné selon

$$\mathcal{S}_{\det}(n,\beta) \le O\tilde{}(n^4\beta + n^3\beta^2). \tag{4}$$

L'algorithme est adaptatif :

Définition 3.1 *Un algorithme est dit adaptatif si son coût dépend des valeurs de ses sorties et pas seulement de bornes au pire sur ces dernières.*

L'algorithme de Clarkson s'adapte à l'ordre de grandeur du déterminant, d'après (3) on voit que dans le cas favorable de matrices n'ayant pas un déterminant relativement «trop petit», c'est-à-dire satisfaisant

$$\log \frac{\prod_{i=1}^{n} \| a_i \|}{\mathrm{vol}(a_1, \ldots a_n)} = O(n) \tag{5}$$

alors l'algorithme s'exécute en seulement

$$O\tilde{}(n^4 + n^3\beta). \tag{6}$$

Puisqu'un nombre de conditionnement associé satisfait à [16, Pb. 13.15] :

$$\log \mathrm{cond}_{\mathrm{det}}(A) = \log \max_{i,j} |a_{i,j}(A^{-1})_{i,j}| \leq \log \frac{\prod_{i=1}^{n} \| a_i \|}{\mathrm{vol}(a_1, \ldots a_n)} + \log \max_{i,j} |a_{i,j}|,$$

on a,

$$\log \mathrm{cond}_{\mathrm{det}}(A) \leq \log \frac{\prod_{i=1}^{n} \| a_i \|}{\mathrm{vol}(a_1, \ldots a_n)} + \beta.$$

Le cas favorable de l'algorithme de Clarkson correspond donc en particulier à des matrices avec ce nombre de conditionnement comme $\log \mathrm{cond}_{\mathrm{det}}(A) = O(n + \beta)$ et non pas comme $O(n\beta)$ dans le pire cas.

Une autre méthode dans la même direction a été proposée par Brönnimann et Yvinec [9]. Leur algorithme de réseau – *lattice algorithm* – généralise, en s'inspirant aussi de Clarkson, la méthode antérieure limitée aux dimensions 2 et 3 de Avnaim *et al.* [3]. On considérera que son coût est analogue à (4).

4 Le déterminant et son signe : calcul multi-modulaire

Une approche multi-modulaire basée sur de nombreux mais petits moduli, permet de combiner des algorithmes adaptatifs et l'utilisation du calcul flottant. L'idée des aspects adaptatifs est de reconstruire le déterminant au fur et à mesure que ses résidus sont calculés. Dès que le déterminant reste stable on peut décréter qu'il est connu. C'est une approche type terminaison anticipée – *early termination* – qui donne des algorithmes probabilistes Monte Carlo. Ceci est à rapprocher des commentaires sur le test d'inversibilité en fin de section 2. On se réfèrera à Brönnimann *et al.* [7] ainsi qu'à [14] pour une analyse de probabilité. Le coût associé est :

$$\mathcal{D}(n, \beta) \leq O\tilde{}(\log |\det(A)| \cdot (n^2\beta + n^3 \log\log\beta) + \log^2 |\det(A)|). \tag{7}$$

La démarche est bien sûr nettement rentable pour les petits déterminants [7, Tables 2 & 3].

Pour le calcul du seul signe du déterminant, les auteurs de [7] propose aussi une version flottante – rapide en pratique – de la méthode de Lagrange pour le théorème chinois. Son coût asymptotique théorique reste borné comme (1).

5 Le déterminant : algorithmique exacte

En arithmétique exacte pour le calcul du déterminant, il existe des résultats allant plus loin – en théorie au moins – que ceux de la section 2.

Une approche adaptative. Avec à l'esprit les règles de Cramer, Pan a proposé dans [21] de calculer le déterminant à partir des dénominateurs de solutions de systèmes linéaires

$$Ax = b, \quad b \text{ vecteur aléatoire}. \tag{8}$$

Puisque la résolution d'un système a une complexité *a priori* moindre que celle du calcul du déterminant, cette technique apporte un gain potentiel. On peut voir les algorithmes de Moenck et Carter [20] ou de Dixon [12] pour la résolution exacte d'un système linéaire. Dû à l'influence de la structure de la matrice (forme normale de Smith), ce gain n'apparaît pas directement dans le pire cas. Cependant une analyse en moyenne ou dans des cas favorables – nous allons le préciser – comme menée par Abbott *et al.* [1] se révèle fructueuse. Ces auteurs procèdent en deux phases. La première consiste à résoudre suffisamment de systèmes linéaires (8) pour trouver un grand diviseur σ du déterminant. La deuxième phase trouve le facteur manquant $\det(A)/\sigma$ par les techniques exactes classiques. Ces deux étapes donnent un coût de

$$\mathcal{D}(n,\beta) \leq O\tilde{\ }\left(n^3\beta^2 + \left(\log \frac{\prod_{i=1}^n \| a_i \|}{|\sigma|}\right) \cdot \left(n^2\beta + n^3\right)\right) \tag{9}$$

qui se ramène au coût (1) dans le pire cas. En conduisant un raisonnement assez analogue à celui de la section 3, on voit que les cas favorables sont ici ceux des matrices telles que σ soit «assez grand». Par exemple pour des matrices aléatoires, des arguments heuristiques [1, Assumption 1] – comparer à (5) – donnent que

$$\log \frac{\prod_{i=1}^n \| a_i \|}{|\sigma|} = O(n). \tag{10}$$

Dans ces cas favorables, le coût donné dans [1] est :

$$O\tilde{\ }(n^4 + n^3\beta^2). \tag{11}$$

D'après la forme normale de Smith. Sans même tirer parti du produit rapide de matrices, l'exposant de n dans le coût du déterminant peut être réduit en deçà de celui qui apparaît dans les formules du type

$$\text{Coût binaire} \ = \ \text{taille du déterminant} \ \times \ \text{coût arithmétique}. \tag{12}$$

que l'on a manipulées jusqu'à présent. Il est possible de tirer profit de la forme normale de Smith (déjà citée plus haut) pour généraliser l'utilisation de systèmes linéaires au calcul du déterminant «complet» sans se limiter au calcul d'un de ses diviseurs [26]. Cette approche – développée initialement pour le calcul du polynôme caractéristique d'une matrice creuse – est approfondie par Eberly *et al.* [13] dans le cas qui nous intéresse. Le résultat est adaptatif. Il dépend de la taille du déterminant et d'un paramètre ϕ caractéristique de la structure de Smith de A qui vérifie

$$\phi = O(\sqrt{|\det(A)|}) \leq O\tilde{\ }(\sqrt{n\beta}).$$

Le coût de l'algorithme est

$$\mathcal{D}(n,\beta) \leq O\tilde{\ }(\phi \cdot n^3\beta^2) \leq O\tilde{\ }(\sqrt{|\det(A)|} \cdot n^3\beta^2) \leq O\tilde{\ }(n^{3.5}\beta^{2.5}). \tag{13}$$

avec une version rapide en

$$\mathcal{D}(n,\beta) \leq O\tilde{\ }(n^2 \cdot \mathcal{M}(\sqrt{n}) \cdot \beta^{1.5}) \leq O(n^{3.19}\beta^{1.5}).$$

On sait aussi qu'en moyenne (voir [13, Corollary 6.3]), le paramètre ϕ est «petit» :

$$\mathsf{E}(\phi) = O(\log n),$$

ce qui donne un coût moyen pour le déterminant de

$$\mathsf{E}(\mathcal{D}(n,\beta)) \leq O\tilde{\ }(n^3\beta^2).$$

On a donc ici un nouveau type de cas favorable qui sont ceux où ϕ «petit». Cette approche est probabiliste Monte Carlo.

D'après un algorithme sans division. Pour «passer outre» la formule (12), l'approche basée sur la forme de Smith s'est focalisée sur le paramètre ϕ de structure de A. L'algorithme au départ algébrique sans division de Kaltofen [18] utilise un autre biais – dit technique «à pas de géant, à pas de bébé» de Shanks (*baby-steps, giant-steps*) – qui consiste directement à réduire la taille des quantités intermédiaires utilisées dans un facteur $\sqrt{n}$. Une analyse de l'algorithme dans le cas entier d'après [17] donne un coût probabiliste Las Vegas de

$$O\tilde{}(\sqrt{n} \cdot (n^3\beta)) \leq O\tilde{}(n^{3.5}\beta), \tag{14}$$

ou encore, si le produit rapide de matrices est utilisé :

$$O(n^{3.07}\beta).$$

L'algorithme de Kaltofen et Villard [19] améliore cette complexité en combinant la technique précédente à un calcul par blocs. Une réduction de la taille dans un facteur $n/\sqrt[3]{n}$ conduit alors à un calcul de déterminant en

$$\mathcal{D}(n, \beta) \leq O\tilde{}(\sqrt[3]{n} \cdot (n^3\beta)) \leq (n^{3.34}\beta). \tag{15}$$

Avec produit rapide de matrices ce coût devient :

$$\mathcal{D}(n, \beta) \leq O(n^{2.7}\beta). \tag{16}$$

Comme pour la méthode de Kaltofen, ceci correspond à un algorithme probabiliste de type Las Vegas.

6 Matrices de nombres flottants

Plutôt que de regarder le comportement du coût relativement à n et à β – surtout en calcul flottant – on peut modéliser la taille des entrées de A selon la taille maximale μ d'une mantisse et la taille maximale η d'un exposant. Les nombres manipulés ensuite au cours des algorithmes sont des flottants creux – *sparse high precision numbers* dans le modèle de Priest [23] – à savoir des listes de couples mantisse / exposant arbitrairement longues. Les opérations arithmétiques sur de tels nombres ont des coûts polynomiaux en leurs tailles.

La question de la complexité du calcul du signe du déterminant se pose alors différemment. Bien sûr, tous les algorithmes précédents ont des coûts exponentiels en la taille η des exposants. On peut s'en convaincre en considérant $\eta = \log \beta$, les fonctions de coût manipulées sont de la forme $n^k(2^\eta)^l$. Remarquons aussi que la méthode naïve de calcul du déterminant comme une somme d'un nombre exponentiel de termes serait polynomiale en η mais essentiellement exponentielle en n. La question se pose donc, voir par exemple Demmel et Koev [11], de savoir s'il est possible de calculer le déterminant au moins avec une bonne précision relative, donc en particulier de calculer son signe, en temps polynomial à la fois en n, en μ et en η ? Même si cette question trouve une réponse positive pour des classes de matrices particulières avec des coefficients non indépendants (voir [11]), le cas général n'est pas résolu.

7 Discussion

Nous réunissons les coûts étudiés dans la table de comparaison ci-dessous. Dans le modèle (discutable) de complexité théorique considéré, et dans les pires cas respectifs, il est difficile de mettre en évidence que le calcul du signe est asymptotiquement moins cher que le calcul du déterminant proprement dit. Il est certainement plus intéressant de discuter des situations où les algorithmes adaptatifs ont un comportement favorable. Une première de ces situations demanderait des analyses complémentaires des cas moyens (ou cas génériques). Ce sera avantageux d'après le conditionnement pour l'algorithme de Clarkson ; d'après les arguments heuristiques de Abbott *et al.* [1] pour leur algorithme ; potentiellement pour la méthode de Lagrange de Brönnimann *et* *al.* [7] ; comme prouvé dans [13] pour l'obtention d'un coût cubique en n. De ces quatres dernières approches, si dans les pire cas tous les termes dominants en n sont linéaires en β, seul l'algorithme de Clarkson conserve un terme linéaire quand la matrice est bien conditionnée (les autres font apparaître un terme quadratique). Là, seul le signe est calculé. Une autres situation favorable est celle de petits déterminants. L'algorithme multi-modulaire de Brönnimann *et al.* est particulièrement rapide dans ce cas.

Table de comparaison des complexités en $O^{\tilde{}}$ $-$ $A \in \mathbb{Z}^{n\times n}$, $\log \| A \|_\infty \leq \beta$,
(avec produit de matrice «lent»).

Méthodes	Pire cas	Cas favorables
Classique $\mathcal{D}(n,\beta)$ cf. (1)	$n^4\beta + n^3\beta^2$	$n^4\beta + n^3\beta^2$
Clarkson $\mathcal{S}_{\det}(n,\beta)$ cf. (3)	$(n^3 + n^2 \log \frac{1}{\Delta(A)}) \cdot (n+\beta)$ $n^4\beta + n^3\beta^2$	$n^4 + n^3\beta$
Multi-modulaire. $\mathcal{D}(n,\beta)$ cf. (7)	$\log\|\det(A)\| \cdot (n^2\beta + n^3 \log\log\beta) + \log\|\det(A)\|)$ $n^4\beta + n^3\beta^2$	$n^2\beta + n^3 \log\log\beta$
Syst. lin. $\mathcal{D}(n,\beta)$ cf. (9)	$n^3\beta^2 + \left(\log \frac{\prod_{i=1}^n \|a_i\|}{\|\sigma\|}\right) \cdot \left(n^2\beta + n^3\right)$ $n^4\beta + n^3\beta^2$	$n^4 + n^3\beta^2$
Smith $\mathcal{D}(n,\beta)$ cf. (13)	$\sqrt{\|\det(A)\|} \cdot n^3\beta^2$ $n^{3.5}\beta^{2.5}$	$\mathsf{E}(\mathcal{D}(n,\beta)) \leq n^3\beta^2$
Sans division $\mathcal{D}(n,\beta)$ cf. (15)	$n^{3.34}\beta$	$n^{3.34}\beta$

Quant à l'exposant de n, on a vu que la formule (12) n'est en rien indicatrice de la complexité. C'est d'abord la technique de Shanks à pas de bébé, à pas de géants appliquée par Kaltofen qui nous le montre [18]. Ses prolongements mènent au coût théorique de $n^{2.7}\beta$ [19]. Ce sont aussi les techniques de Villard [26] et de Eberly *et al.* [13] avec le plus faible coût en moyenne.

Les directions qui s'ouvrent sont de mieux comprendre l'agencement des différents cas favorables et d'étudier spécifiquement le coût du calcul du signe avec les approches aux faibles exposants de n.

8 Conclusion

On peut conclure simplement en dressant la liste de certains points qui n'ont pas été du tout abordés et constituent un point faible de l'étude :

- facteurs logarithmiques masqués par les $O^{\tilde{}}$;
- coûts en utilisation mémoire ;
- étude en pratique ;
- distinction déterministe, probabiliste, heuristique ;
- cas de matrices structurées et creuses ;
- problèmes voisins, résolution de systèmes linéaires ou le polynôme caractéristique ;
- domaines de calcul autres que les nombres flottants et les entiers.

Références

[1] Abbott, J., Bronstein, M., and Mulders, T. Fast deterministic computation of determinants of dense matrices. In *International Symposium on Symbolic and Algebraic Computation, Vancouver, BC, Canada* (Jul 1999), pp. 197–204.

[2] Allender, E., Beals, R., and Ogihara, M. The complexity of matrix rank and feasible systems of linear equations. *Computational Complexity 8*, 2 (1999), 99–126.

[3] Avnaim, F. and Boissonnat, J.D. and Devillers, O. and Preparata, F. and Yvinec, M. Evaluating signs of determinants using single-precision arithmetic. *Algorithmica 17* (1997), 111–132.

[4] Bareiss, E. Computational solution of matrix problems over an integral domain. *J. Inst. Math. Appl. 10* (1972), 68–104.

[5] Baur, W., and Strassen, V. The complexity of partial derivatives. *Theor. Comp. Sc. 22* (1982), 317–330.

[6] Brönnimann, H., Emiris, I., Pan, V., and Pion, S. Computing exact geometric predicates using modular arithmetic. In *Proc. 13th Annual ACM Symp. Comput. Geom.* (1997), pp. 174–182.

[7] Brönnimann, H., Emiris, I., Pan, V., and Pion, S. Sign determination in residue number systems. *Theoret. Comput. Sci. 210*, 1 (1999), 173–197. Special Issue on Real Numbers and Computers.

[8] Brönnimann, H. and Yvinec, M. A complete analysis of Clarkson's algorithm for safe determinant evaluation. Tech. rep., Institut National de Recherche en Informatique et en Automatique, INRIA-2051, Novembre 1996.

[9] Brönnimann, H. and Yvinec, M. Efficient exact evaluation of signs of determinant. *Algorithmica 27* (2000), 21–56.

[10] Clarkson, K. Safe and effective determinant evaluation. In *Proc. 33rd IEEE Symp. Foundations Of Computer Science, Pittsburg, USA* (1992).

[11] Demmel, J., and Koev, P. Necessary and sufficient conditions for accurate and efficient rational function evaluation and factorizations of rational matrices. Submitted, 2000.

[12] Dixon, J. Exact solution of linear equations using p-adic expansions. *Numer. Math. 40* (1982), 137–141.

[13] Eberly, W., Giesbrecht, M., and Villard, G. Computing the determinant and Smith form of an integer matrix. In *The 41st Annual IEEE Symposium on Foundations of Computer Science, Redondo Beach, CA* (Nov. 2000).

[14] Emiris, I. A complete implementation for computing general dimensional convex hulls. *Inter. J. Computational Geometry & Applications 8*, 2 (1998), 223–253.

[15] von zur Gathen, J., and Gerhard, J. *Modern Computer Algebra.* Cambridge University Press, 1999.

[16] Higham, N. *Accuracy and stability of numerical algorithms.* SIAM, Philadelphia, 1996.

[17] Kaltofen, E. Communication personnelle, 2000.

[18] Kaltofen, E. On computing determinants without divisions. In *International Symposium on Symbolic and Algebraic Computation, Berkeley, California USA* (July 1992), ACM Press, pp. 342–349.

[19] Kaltofen, E., and Villard, G. On the complexity of computing determinants, Feb. 2001. In preparation.

[20] Moenck, R., and Carter, J. Approximate algorithms to derive exact solutions to systems of linear equations. In *Proc. EUROSAM* (1979), LNCS 72, Springer Verlag, pp. 63–73.

[21] Pan, V. Computing the determinant and the characteristic polynomial of a matrix via solving linear systems of equations. *Inf. Proc. Letters 28* (1988), 71–75.

[22] Pion, S. De la géométrie algorithmique au calcul géométrique. Thèse de Doctorat, Université de Nice Sophia-Antipolis, France, nov. 1999.

[23] Priest, D. Algorithms for arbitrary precision floating point arithmetic. In *Proceedings of the 10th Symposium on Computer Arithmetic* (jun 1991), IEEE Computer Society Press, pp. 132–145.

[24] Strassen, V. Gaussian elimination is not optimal. *Numer. Math. 13* (1969), 354–356.

[25] Strassen, V. Vermeidung von divisionen. *J. Reine Angew. Math. 264* (1973), 182–202.

[26] Villard, G. Computing the Frobenius form of a sparse matrix. In *The Third International Workshop on Computer Algebra in Scientific Computing, Samarkand, Uzbekistan* (Oct. 2000).